ATLAS
MANUAL DE USUARIO
COMPONENTE INPUT IBAN

Versión 1.1

Área de Aplicaciones Especiales y Arquitectura de Software

Agencia de Informática y Comunicaciones de la Comunidad de Madrid

Hoja de Control

Título	Manual de usuario Componente Input Iban		
Documento de Referencia	NORMATIVA ATLAS		
Responsable	Área de Aplicaciones Especiales y Arquitectura de Software		
Versión	1.1	**Fecha Versión**	30/04/2014

Registro de Cambios

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Versión inicial del documento	Área de Integración y Arquitectura de Aplicaciones	30/04/2014
1.1	Apartado 4.4: • Se han añadido 2 atributos: *readonly* y *disabled*	Área de Integración y Arquitectura de Aplicaciones	03/07/14

Índice

1. INTRODUCCIÓN 4

1.1. AUDIENCIA OBJETIVO 4
1.2. CONOCIMIENTOS PREVIOS 4

2. DESCRIPCIÓN 5

3. INSTALACIÓN Y CONFIGURACIÓN 7

3.1. INSTALACIÓN 7
3.2. CONFIGURACIÓN 7

4. USO 8

4.1. PASO 1: DEFINICIÓN DEL ESPACIO DE NOMBRES DE ETIQUETAS DE ATLAS COMPOSITE 8
4.2. PASO 2: INSERCIÓN EN LA PÁGINA DE LA ETIQUETA DE <ATLASC:INPUTIBAN> 9
4.3. PASO 3: CREACIÓN DEL BEAN DE RESPALDO 10
4.4. ATRIBUTOS 11
4.5. CLASES CSS 14
4.6. API JAVASCRIPT 15
4.7. EJEMPLO DE USO 16
4.8. USO SIN LA PARTE VISUAL 17
4.8.1. `Iban` 17
4.8.1. `IbanFormatException` 21
4.8.1. `ICodigoPaisValidator` 21
4.8.1. `CodigoPaisValidator` 21

5. PREGUNTAS MÁS FRECUENTES 23

6. ENLACES RELACIONADOS 24

1. INTRODUCCIÓN

Este documento contiene el manual de uso del componente visual ***Input IBAN*** del Framework Atlas. En él se incluye información sobre cómo utilizar dicho componente en una aplicación Web, así como información acerca de la configuración de los parámetros fundamentales del componente.

1.1. AUDIENCIA OBJETIVO

Este documento está orientado a toda aquella persona que esté desarrollando una aplicación Web basada en el Framework Atlas y necesite utilizar componentes de presentación en su aplicación Web.

1.2. CONOCIMIENTOS PREVIOS

Para un completo entendimiento del documento, el lector deberá tener conocimientos previos sobre las siguientes tecnologías:

- JavaServer Faces Technology (JSF)
- Javascript
- CSS

Para saber más sobre dichas tecnologías, consultar el apartado de este documento, ver apartado ENLACES RELACIONADOS.

2. DESCRIPCIÓN

Este componente combina una serie de componentes visuales y lógica de validación para que el usuario introduzca un Código Internacional de Cuenta Bancaria (IBAN) válido en la aplicación. Para la representación de un objeto IBAN se ha creado la clase `Iban` (Ver a apartado Iban).

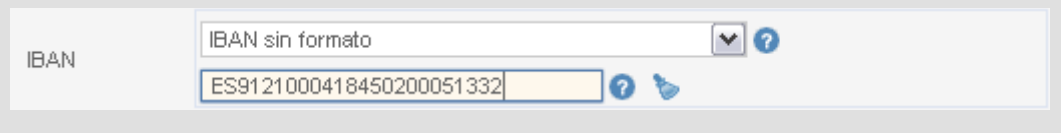
Dispone de tres modos de funcionamiento y de numerosas opciones para poder adecuarlo a las necesidades de cada aplicación. Aun así, el componente ya viene configurado para empezar a usarlo sin especificar, a penas, ningún parámetro.

El modo (atributo *mode*) del componente Input Iban sirve para establecer el funcionamiento del propio componente y está pensado para contemplar casos específicos en los que un IBAN requiera una apariencia visual o una validación propias. Los modos disponibles son:

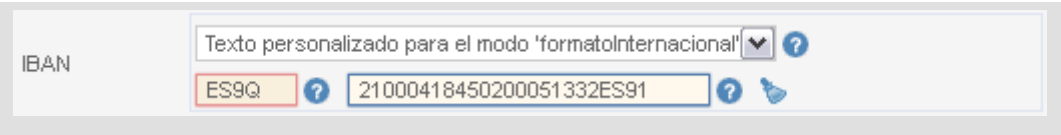
- Sin formato (*mode=sinFormato*): se compone de un único campo de texto
- Formato internacional (*mode=formatoInternacional*): Se compone de un campo de texto para escribir el código de país y los dígitos de control del iban y otro para el código de cuenta nacional
- Formato nacional (*mode=formatoNacional*): Al igual que en el formato anterior, dispone de un primer campo con el país y los dígitos del control del IBAN, pero en este caso la cuenta nacional se descompone en cuatro campos al estilo de las cuentas bancarias españolas (CCC)**. Este modo valida que sea una cuenta bancaria española válida**.

Básicamente el componente, dispone de una lista desplegable para elegir el modo de funcionamiento del componente y en la siguiente línea los campos de texto específicos de cada modo.

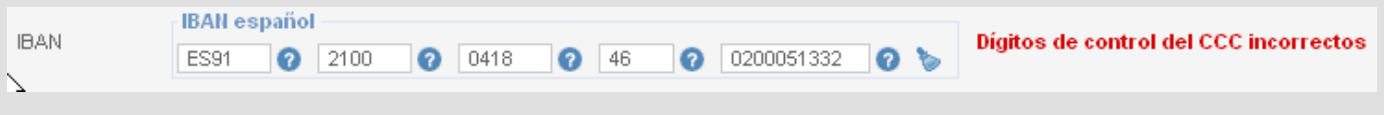
A continuación se muestra un ejemplo del componente con el modo *Sin Formato*.



Seguidamente el modo *Formato Internacional*. En este ejemplo se ha personalizado la etiqueta del modo *Formato Internacional* a través del atributo *formatoInternacionalModeLabel*. Como veremos más adelante, el componente dispone de validaciones de formato en cliente, así, observamos que el primer campo está “marcado” de rojo ya que tiene un formato incorrecto (deberían ser 2 letras mayúsculas para el país + 2 dígitos de control).



Por último se muestra un ejemplo del modo Formato Nacional. Esta vez: se ha ocultado la lista de selección de modo (atributo *showModeSelector="false"*); se ha especificado el modo *Formato Nacional* por defecto (atributo *defaultMode="formatoNacional"*); y se ha indicado un texto para la cabecera (atributo *header="IBAN español"*).



Como se puede apreciar, el IBAN introducido no tiene ningún campo marcado en rojo -ya que todos tienen un formato correcto- pero al realizar la validación en servidor del número de cuenta, se comprueba que se corresponde con una cuenta española con los dígitos de control incorrectos, por lo que lanza un error.

Todas las validaciones del componte se realizan en el servidor. Cuando se produzca un error de validación el componente añadirá un `FacesMessage` con un mensaje de resumen (`FacesMessage#getSummary()`) y otro más detallado (`FacesMessage#getDetail()`) de la causa del error. Los mensajes de error se adecuarán al modo seleccionado en el componente, de manera que, el usuario perciba fácilmente dónde y por qué está fallando la validación.

Como ya se ha ido adelantando, la parte cliente del componente se ha enriquecido, tanto en estilo como en usabilidad, añadiendo CSS y JavaScript. Algunas de las características añadidas son:

- Estilo propio para el campo que tiene el foco
- Estilo propio para los campos que contengan datos con formato incorrecto
- Paso automático al siguiente campo cuando el actual esté relleno
- Traspaso automático de los datos al cambiar de modo

Todas las clases CSS que son utilizadas en el componente son explicadas posteriormente en el apartado CLASES CSS.

De la misma forma, también existe en el apartado API JAVASCRIPT donde se explica las funciones de cliente disponibles para el componente.

3. INSTALACIÓN Y CONFIGURACIÓN

En este apartado se incluye información sobre la instalación y la configuración del componente *Input IBAN.*

3.1. INSTALACIÓN

El componente Input Iban ya viene instalado en el arquetipo Web, incluido con el módulo de componentes visuales. Por este motivo no es necesaria una instalación adicional si se parte del arquetipo.

3.2. CONFIGURACIÓN

No es necesaria ninguna configuración adicional si se ha partido del arquetipo web.

4. USO

Una vez instalado el módulo de componentes puede procederse a su utilización. Para ello deben realizarse los pasos indicados en los siguientes apartados:

4.1. Paso 1: Definición del espacio de nombres de etiquetas de Atlas Composite

Es necesario crear un fichero *xhtml* y establecer la definición del espacio de nombres para las etiquetas de componentes de Atlas Composite. Un ejemplo de cabecera de fichero *xhtml* es la siguiente:

Cabecera de fichero xhtml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:atlasc="http://atlas.core.componentes/jsf/composite"
      xmlns:a4j="http://richfaces.org/a4j">
```

4.2. Paso 2: Inserción en la página de la etiqueta de `<atlasc:inputIban>`

Se incluirá una etiqueta **`<atlasc:inputIban>`** que inserta el componente en la página. Cuando se realice una petición en la que esté incluido el componente, éste se validará y en el caso de fallar, lanzará una excepción de validación de JSF con el mensaje de error. Por lo tanto, es necesario que en la página haya un componente para mostrar los errores.

A continuación se muestra un ejemplo:

InputIbanSample.xhtml

```
<h:panelGrid columns="2" cellpadding="0" cellspacing="0">

      <h:outputLabel value="IBAN" for="iban" />
      <h:panelGrid cellpadding="0" cellspacing="0" columns="2"
            styleClass="componenteConMsj">
            <atlasc:inputIban id="iban" value="#{inputIbanSampleBean.iban}" />
            <rich:message for="iban" styleClass="errorRich"
                  showSummary="true" showDetail="false" />
      </h:panelGrid>

</h:panelGrid>
```

Técnicamente, el componente se ha creado través de un composite que hereda del componente más básico de entrada de JSF, `UIInput`. Por lo que la manera de usar el componente es similar a la de cualquier componente JSF que herede de este, véase `<h:inputText>` o `<h:selectOneMenu>` entre otros. Así, por ejemplo se le podría anidar un validador propio que extendiera el que ya viene por defecto.

La manera de mostrar los mensajes de error es la misma que para cualquier componente JSF. Teniendo en cuenta que las librerías de componentes disponibles en Atlas, las posibilidades son:

- `<h:message>` y `<rich:message>` indicando en el atributo *for* el *id* del componente `<atlasc:inputIban>`
- o `<h:messages>` y `<rich:messages>` con el parámetro *globalOnly=false* (ya que el mensaje JSF que encola el componente viene asociado a su id de cliente)

Como se explicó en la descripción, el componente encola un mensaje JSF con un resumen y un detalle del error. Para elegir el mensaje que se quiere mostrar los componentes `<h:message/s>` y `<rich:message/s>` disponen de los siguientes atributos *showDetail* y *showSummary*.

La lista completa de atributos disponibles para el componente se encuentra en la sección ATRIBUTOS.

4.3. Paso 3: Creación del Bean de Respaldo

Será necesario declarar una propiedad de tipo **`Iban`** *(`atlas.banca.iban.Iban`)* para utilizarla en el atributo *value* del componente (una para cada componente `<atlasc:inputIban>`).

InputIbanSample.java

```
package atlas.samples.bean;
import atlas.banca.iban.Iban;

@ManagedBean(name="inputIbanSampleBean")
@RequestScoped
public class InputIbanSampleBean {

    private Iban iban;

    public Iban getIban() {
        return iban;
    }

    public void setIban(Iban iban) {
        this.iban = iban;
    }

}
```

4.4. ATRIBUTOS

El Backing Component que maneja el tag `<atlasc:inputIban>` es `atlas.componentes.composite.InputIban` cuyos atributos se describen a continuación:

Nombre	Obligatorio	Tipo	Descripción
id	NO	`String`[1]	Identificador del componente
value	SI	`Iban`[2]	Valor del componente
styleClass	NO	`String`	Clase/s CSS del contenedor del componente
defaultMode	NO	`Mode`[3]	Modo por defecto. Los posibles valores son: 'sinFormato', 'formatoInternacional' o 'formatoNacional'. Por defecto 'formatoNacional'
showModeSelector	NO	`Boolean`[4]	Indica si debe mostrarse la selección de modo o no. Por defecto true
sinFormatoModeAvailable	NO	`Boolean`	Indica si el modo 'IBAN sin formato' está disponible o no. Por defecto 'true'
formatoInternacionalModeAvailable	NO	`Boolean`	Indica si el modo 'IBAN con formato internacional' está disponible o no. Por defecto 'true'
formatoNacionalModeAvailable	NO	`Boolean`	Indica si el modo 'IBAN con formato español' está disponible o no. Por defecto 'true'
sinFormatoModeLabel	NO	`String`	Texto que se mostrará en la selección del modo 'sinFormato'. Por defecto 'IBAN sin formato'
formatoInternacionalModeLabel	NO	`String`	Texto que se mostrará en la selección del modo 'formatoInternacional'. Por defecto 'IBAN con formato internacional'

formatoNacionalModeLabel	NO	`String`	Texto que se mostrará en la selección del modo 'formatoNacional'. Por defecto 'IBAN con formato español (sólo cuentas nacionales)'
header	NO	`String`	Texto que se mostrará en la cabecera del componente. Por defecto no se muestra nada
immediate	NO	`Boolean`	Indica si la validación del componente debe adelantarse a la fase 'Apply Request Values' o no. Valor por defecto false
required	NO	`Boolean`	Valor true|false para indicar si es obligatorio introducir un IBAN válido o no. El valor por defecto es false
requiredMessage	NO	`String`	Texto a utilizar para el mensaje en caso en que sea obligatorio introducir un IBAN válido y no se introduzca. Por defecto se muestra 'Introduzca un IBAN válido'
validator	NO	`MethodBinding`[5]	MethodExpression que representa un método de validación que se llamará durante la fase Process Validations para llevar a cabo las comprobaciones de validez del valor del componente. La expresión debe evaluar un método público que tenga los parámetros FacesContext, UIComponent, y Object, y que devuelva un void
validatorMessage	NO	`String`	Mensaje que se mostrará cuando el validator no de por válido el valor introducido
disabled	NO	`Boolean`	Si 'true'. El componente será deshabilitado. Por defecto 'false'
readonly	NO	`Boolean`	Si 'true'. Los campos de texto serán de sólo lectura. Por defecto 'false'

Referencias:

1. `java.lang.String`
2. `atlas.banca.iban.Iban`
3. `atlas.componentes.composite.InputIban.Mode`
4. `java.lang.Boolean`
5. `javax.faces.el.MethodBinding`

4.5. CLASES CSS

Clases CSS utilizadas para definir el estilo del componente:

Clase (selector)	Descripción
.atlas-iban	Clase que define el estilo del contenedor del componente
.atlas-iban-hdr	Clase que define el estilo la cabecera del componente
.atlas-iban-hdr-lbl	Clase que define el estilo del texto de la cabecera del componente
.atlas-iban-modo	Clase que define el estilo del contenedor de la lista desplegable de modos
.atlas-iban-modo-slc	Clase que define el estilo de la lista desplegable de modos
.atlas-iban-cnt	Clase que define el estilo del contenedor de los diferentes modos
.atlas-iban-cnt-sf	Clase que define el estilo del contenedor del modo sin formato
.atlas-iban-cnt-sf-iban	Clase que define el estilo del campo iban en el modo sin formato
.atlas-iban-cnt-fi	Clase que define el estilo del contenedor del modo formato internacional
.atlas-iban-cnt-fi-pdc	Clase que define el estilo del primer campo (país + dígitos de control) en el modo formato internacional
.atlas-iban-cnt-fi-bban	Clase que define el estilo del segundo campo (cuenta nacional) en el modo formato internacional
.atlas-iban-cnt-fn	Clase que define el estilo del contenedor del modo formato nacional
.atlas-iban-cnt-fn-pdc	Clase que define el estilo del primer campo (pais + dígitos de control) en el modo formato nacional
.atlas-iban-cnt-fn-ent	Clase que define el estilo del segundo campo (banco o entidad) en el modo formato nacional
.atlas-iban-cnt-fn-ofi	Clase que define el estilo del tercer campo (sucursal u oficina) en el modo formato nacional
.atlas-iban-cnt-fn-num	Clase que define el estilo del cuarto campo (número) en el modo formato nacional

4.6. API JAVASCRIPT

El componente `<atlasc:inputIban>` también puede ser controlado a través de su API JavaScript. Se proporcionan las siguientes funciones:

Nombre	Descripción
limpiaCampos()	Limpia todos los campos del componente

4.7. EJEMPLO DE USO

Se puede ver un ejemplo del componente, tanto de su uso básico como de la configuración avanzada integrado en la aplicación de demostración de componentes en la siguiente ruta:

Inicio > Formularios > Input IBAN

4.8. USO SIN LA PARTE VISUAL

Para la representación del objeto IBAN se ha creado la clase `atlas.banca.iban.Iban`, esta clase ya se encuentra incorporada en las aplicaciones generadas a partir de los arquetipos web de Atlas.

4.8.1. `Iban`

Esta clase representa un **Código Internacional de Cuenta Bancaria** (IBAN, International Bank Account Number) codificado como se especifica en el estándar ISO 13616:2007.

El IBAN consta de un máximo de 34 caracteres alfanuméricos. Los dos primeros son de carácter alfabético e identifican el país. Los dos siguientes son dígitos de control y constituyen el elemento de validación de la totalidad del IBAN. Los restantes son el BBAN (Basic Bank Acount Number), propios de cada país y se suele componer del número de cuenta, que, en la mayoría de los casos, identifica además la entidad y la oficina.

En el caso español están los cuatro primeros, formados por los elementos explicados en el párrafo anterior, y luego se sigue con los 20 caracteres numéricos actuales del Código Cuenta Cliente (CCC) -es decir, consta de un total de 24 caracteres-. Dichos cuatro nuevos caracteres corresponden a:

Ejemplos: comparativa entre el código cuenta cliente (CCC) y código IBAN de una misma cuenta.

- Código Cuenta Cliente (C.C.C.): 2077 0024 00 3102575766
- Código IBAN:
  - Formato papel: IBAN ES76 2077 0024 0031 0257 5766
  - Formato electrónico: ES7620770024003102575766

Esta clase valida lo siguiente:

- Formato básico correcto
- Dígitos de control correctos
- En el caso que sea una cuenta española (País='ES') se validará el correcto formato del CCC como los dígitos de control

Atención
Esta clase no valida si los códigos de país utilizado son realmente válidos por la ISO 3166-1, simplemente validamos que esté formado por 2 letras mayúsculas. Así mismo tampoco se valida que el BBAN introducido tenga una estructura correcta (a excepción de España).

Las instancias de esta clase son seguras e inmutables, listas para ser usadas por múltiples hilos

A continuación se detallan los constructores y métodos que dispone la clase:

4.8.1.1. Constructores

Iban.java

```
/**
 * Crea un IBAN a partir de la cadena pasada
 *
 * @param iban un String no nulo
 * @throws IbanFormatException si el String pasado no es un IBAN válido
 */
public Iban(String iban) throws IbanFormatException;

/**
 * Crea un nuevo IBAN a partir del primer grupo del IBAN (4 dígitos) y de
 * la cuenta nacional (BBAN)
 *
 * @param primerGrupo Se corresponde con los primeros 4 dígitos de un IBAN:
 *            2 para el código del país y 2 para los dígitos del control.
 * @param cuentaNacional Una cadena no nula
 * @throws IbanFormatException si no se puede crear un IBAN válido
 */
public Iban(String primerGrupo, String cuentaNacional) throws
   IbanFormatException;

/**
 * Crea un nuevo IBAN a partir del código de país, de los dígitos de
 * control y de la cuenta nacional (BBAN)
 *
 * @param pais Código alfa-2 que identifican al país en la ISO 3166-1.
 * @param dc son los dígitos de control del IBAN
 * @param cuentaNacional Una cadena no nula que identifica la cuenta
 *            bancaria dentro de cada país
 * @throws IbanFormatException si no se puede crear un IBAN válido
 */
public Iban(String pais, String dc, String cuentaNacional) throws
    IbanFormatException;

/**
 * Crea un nuevo IBAN español a partir de todas las partes que lo componen
 *
 * El IBAN creado tendrá el siguiente formato:
 *     - primerGrupo + entidad + oficina + dcCCC + numeroCuenta
 *
 * @param primerGrupo se corresponde con los primeros 4 dígitos de un IBAN
 *            español: 2 para el código de España 'ES' y 2 para los dígitos
 *            del control del IBAN. Por Ej.: 'ES33'
 * @param entidadCCC entidad bancaria del CCC
 * @param oficinaCCC oficina bancaria del CCC
 * @param dcCCC digitos de control del CCC
 * @param numeroCuentaCCC del CCC
 * @throws IbanFormatException Si hay algún error de formato
 */
public Iban(String primerGrupo, String entidadCCC, String oficinaCCC,
 String dcCCC, String numeroCuentaCCC) throws IbanFormatException;
```

4.8.1.1. Métodos de clase

Iban.java

```
/**
 * Extrae el código de país de este IBAN
 *
 * @return Una cadena no nula con el código de país del IBAN
 */
public String getCodigoPais();

/**
 * Extrae los dígitos de control de este IBAN
 *
 * @return Una cadena no nula con los dígitos de control
 */
public String getDigitosControl();

/**
 * Extrae la cadena que representa el BBAN (cuanta nacional) de este IBAN
 *
 * @return Una cadena que representa el BBAN de este IBAN
 */
public String getBban();

/**
 * Devuelve el IBAN en formato papel, listo para ser impreso.
 *
 * Cuando va a ser impreso el IBAN es expresado en grupos de 4 caracteres
 * separados por un espacio, siendo el último grupo de longitud variable.
 *
 * @return Una cadena no nula con este IBAN en formato papel
 */
@Override
public String toFormatoPapel();
```

4.8.1.1. Métodos estáticos de ayuda

Iban.java

```
/**
 * Comprueba que la cadena pasada sea un IBAN válido
 *
 * Para validar el código de país se realiza la comprobación de que sean 2
 * letras mayúsculas, puede personalizar dicha comprobación a través del
 * método {@link Iban#isValido(String, ICodigoPaisValidator)}
 *
 * @param iban un String.
 * @return true si la cadena es un IBAN válido, false en
 *         otro caso.
 */
public static boolean isValido(String iban);

/**
 * Comprueba que la cadena pasada sea un IBAN válido
 *
 * Con el parámetro validadorPais puede crear un
 * validador de códigos de países a medida para su aplicación
 *
 * @param iban un String.
 * @param validadorPais validador del país personalizado
 * @return true si la cadena es un IBAN válido, false en
 *         otro caso.
 */
public static boolean isValido(String iban, ICodigoPaisValidator
  validadorPais);

/**
 * Devuelve una cadena con la representación en formato electrónico del
 * IBAN formado a partir del código de país y el BBAN pasados como
 * parámetros
 *
 * Cuando va a ser utilizado electrónicamente el IBAN es expresado con
 * todos los caracteres juntos, sin espacios y en mayúsculas.
 *
 * @param pais código de país
 * @param bban código de cuenta identificativo de cada país (Basic Bank
 *            Account Number)
 * @return Una cadena no nula con el IBAN en formato papel
 * @throws IbanFormatException si se ha producido un error de formato al
 *             crear el BBAN
 */
public static String toFormatoElectronico(String pais, String bban)
  throws IbanFormatException;

/**
 * Devuelve una cadena con la representación en formato papel del IBAN
 * formado a partir del código de país y el BBAN pasados como parámetros
 *
 * Cuando va a ser impreso el IBAN es expresado en grupos de 4 caracteres
 * separados por un espacio, siendo el último grupo de longitud variable.
 *
 * @param pais código de país (alfa-2 de la ISO 3166-1)
 * @param bban código de cuenta identificativo de cada país (Basic Bank
 *            Account Number)
 * @return Una cadena no nula con el IBAN en formato papel
 * @throws IbanFormatException si se ha producido un error de formato al
 *             crear el IBAN
 */
public static String toFormatoPapel(String pais, String bban)
  throws IbanFormatException;
```

4.8.1. IbanFormatException

Excepción creada para ser lanzada cuando en un intento de convertir una cadena en un `Iban` se produzca un error.

IbanFormatException.**java**

```
/**
 * Devuelve la cadena que causó el error
 *
 * @return cadena que causó el error
 */
public String getInputString();

/**
 * Devuelve un mensaje detallado del error
 *
 * @return mensaje de error
 */
public String getMsjDetalle();

/**
 * Devuelve un mensaje resumen del error
 *
 * @return mensaje de error
 */
@Override
public String getMessage();
```

4.8.1. ICodigoPaisValidator

Interfaz que declara el método que tienen que implementar las clases validadoras de código de país utilizadas en el método Métodos estáticos de ayuda.

ICodigoPaisValidator.**java**

```
/**
 * Valida que la cadena pasada por parámetro se corresponda con un código
 * de país válido
 *
 * @param codigoPais cadena con el código de país
 * @return true si es la cadena representa un país válido,
 *         false en otro caso
 */
public boolean valida(String codigoPais);
```

4.8.1. CodigoPaisValidator

Clase singleton validadora del código de país utilizada en la clase `Iban`, tanto en la creación de nuevos objetos como en el método Métodos estáticos de ayuda. Esta clase implementa la anterior interfaz `ICodigoPaisValidator`.

CodigoPaisValidator.**java**

```
/**
 * Valida que la cadena pasada por parámetro se corresponda con un código
 * de país válido.
 *
 * La validación que realiza es que la cadena sean dos letras mayúsculas.
 *
 * @param codigoPais cadena con el código de país
 * @return true si es la cadena representa un país válido,
 *         false en otro caso
 */
@Override
public boolean valida(String codigoPais);
```

5. PREGUNTAS MÁS FRECUENTES

En este apartado se incluyen una lista de preguntas más frecuentes sobre el componente.

Pregunta: ¿Dónde puedo encontrar información general sobre los componentes?
Respuesta: En la aplicación de demostración de los componentes del Framework Atlas

Pregunta: ¿Cómo se ha implementado el componente en su capa de presentación?
Respuesta: Se ha utilizado la tecnología Composite de JSF, Facelets, JacaScript, Jquery y CSS.

Pregunta: ¿Cómo puedo modificar los estilos del componente?
Respuesta: Mediante los atributos de estilo del componente. Para más información consultar la tabla de atributos en la sección de *Uso* del componente.

Pregunta: ¿Cómo se muestran los mensajes de error?
Respuesta: Los mensajes se añaden al contexto de JSF, para mostrarlos será necesario que la página contenga un componente de tipo h:messages, ya que el componente número de documento no lo incluye.

6. ENLACES RELACIONADOS

Producto	URL
JavaServer Faces Technology	http://docs.oracle.com/javaee/6/tutorial/doc/bnaph.html
MyFaces	http://myfaces.apache.org/
RichFaces	http://www.jboss.org/richfaces
JavaScript	https://developer.mozilla.org/es/docs/JavaScript
Jquery	http://api.jquery.com/
CSS	http://www.w3schools.com/css/default.asp
IBAN	http://es.wikipedia.org/wiki/International_Bank_Account_Number